ECE4530 Fall 2019: Handout Lecture 9

reference software

```
unsigned long long mymul(unsigned long a, unsigned long b) {
   unsigned long long r;
   r = (unsigned long long) a * b;
   return r;
}
```

reference software (assembly)

```
;    TimerLap();
;    sw_tt = mymul(arga, argb);
;    sw_ct = TimerLap();
;
; is implemented as:

    f922: 84 12         call  r4        ;         call TimerLap()

    f924: 18 41 0e 00   mov 14(r1), r8  ;0x0000e  prepare arg for __mspabi_mpyll
    f928: 19 41 10 00   mov 16(r1), r9  ;0x00010  prepare arg for __mspabi_mpyll
    f92c: 1c 41 12 00   mov 18(r1), r12 ;0x00012  prepare arg for __mspabi_mpyll
    f930: 1d 41 14 00   mov 20(r1), r13 ;0x00014  prepare arg for __mspabi_mpyll
    f934: 4e 43         clr.b r14   ;             prepare arg for __mspabi_mpyll
    f936: 0f 4e         mov r14,  r15 ;           prepare arg for __mspabi_mpyll
    f938: 0a 4e         mov r14,  r10 ;           prepare arg for __mspabi_mpyll
    f93a: 0b 4e         mov r14,  r11 ;           prepare arg for __mspabi_mpyll
    f93c: b0 12 ae fa   call  #-1362    ;#0xfaae  call __mspabi_mpyll

    f940: 08 4c         mov r12,  r8  ;           prepare return
    f942: 09 4d         mov r13,  r9  ;           prepare return
    f944: 0a 4e         mov r14,  r10 ;           prepare return
    f946: 81 4f 02 00   mov r15,  2(r1) ;         prepare return

    f94a: 84 12         call  r4    ;             call TimerLap()
```

hardware multiply coprocessor

```
module  mymul (
              output [15:0] per_dout,   input         mclk,
              input [13:0]  per_addr,   input [15:0]  per_din,
              input         per_en,     input [1:0]   per_we,
              input         puc_rst
              );
   reg [31:0]               hw_a, hw_b;
   reg [63:0]               hw_retval;
   reg                      hw_ctl, hwctl_old;
   wire [63:0]              mulresult;
   wire                     write_alo, write_ahi,
                            write_blo, write_bhi,
                            write_retval, write_ctl;
   always @(posedge mclk or posedge puc_rst)
     if (puc_rst)
       begin
          hw_a        <= 32'h0;
          hw_b        <= 32'h0;
          hw_retval   <= 64'h0;
          hw_ctl      <= 1'h0;
          hw_ctl_old  <= 1'h0;
       end
     else
       begin
          hw_a[15: 0] <= write_alo    ? per_din[15:0]   : hw_a[15: 0];
          hw_a[31:16] <= write_ahi    ? per_din[15:0]   : hw_a[31:16];
          hw_b[15: 0] <= write_blo    ? per_din[15:0]   : hw_b[15: 0];
          hw_b[31:16] <= write_bhi    ? per_din[15:0]   : hw_b[31:16];
          hw_retval   <= write_retval ? mulresult       : hw_retval;
          hw_ctl      <= write_ctl    ? per_din[0]      : hw_ctl;
          hw_ctl_old  <= hw_ctl;
       end

   assign mulresult = hw_a * hw_b;

   assign write_alo    = (per_en & (per_addr == 14'hA0) & (per_we == 2'h3));
   assign write_ahi    = (per_en & (per_addr == 14'hA1) & (per_we == 2'h3));
   assign write_blo    = (per_en & (per_addr == 14'hA2) & (per_we == 2'h3));
   assign write_bhi    = (per_en & (per_addr == 14'hA3) & (per_we == 2'h3));
   assign write_ctl    = (per_en & (per_addr == 14'hA8) & per_we[0] & per_we[1]);
   assign write_retval = ((hw_ctl == 1'h1) & (hw_ctl ^ hw_ctl_old));
   assign per_dout = (per_en & (per_addr == 14'hA4) & (per_we == 2'h0)) ? hw_retval[15: 0] :
                     (per_en & (per_addr == 14'hA5) & (per_we == 2'h0)) ? hw_retval[31:16] :
                     (per_en & (per_addr == 14'hA6) & (per_we == 2'h0)) ? hw_retval[47:32] :
                     (per_en & (per_addr == 14'hA7) & (per_we == 2'h0)) ? hw_retval[63:47] : 16'h0;
endmodule
```

software driver

```
#define HW_A      (*(volatile unsigned long *)      0x140)
#define HW_B      (*(volatile unsigned long *)      0x144)
#define HW_RETVAL (*(volatile unsigned long long *) 0x148)
#define HW_CTL    (*(volatile unsigned *)           0x150)

unsigned long long mymul_hw(unsigned long a, unsigned long b) {
  HW_A = a;
  HW_B = b;
  HW_CTL = 1;
  HW_CTL = 0;
  return HW_RETVAL;
}
```

software driver (assembly)

```
;    TimerLap();
;    hw_tt = mymul_hw(arga, argb);
;    hw_ct = TimerLap();
;
; is implemented as:

    f950: 84 12         call  r4    ;              call TimerLap()
    f952: 1c 41 0e 00   mov 14(r1), r12 ;0x0000e   prepare arg for mymul_hw
    f956: 1d 41 10 00   mov 16(r1), r13 ;0x00010   prepare arg for mymul_hw
    f95a: 1e 41 12 00   mov 18(r1), r14 ;0x00012   prepare arg for mymul_hw
    f95e: 1f 41 14 00   mov 20(r1), r15 ;0x00014   prepare arg for mymul_hw
    f962: b0 12 46 f8   call  #-1978    ;#0xf846   call mymul_hw
    f966: 81 4c 08 00   mov r12,  8(r1) ;          prepare return
    f96a: 81 4d 0a 00   mov r13,  10(r1)  ; 0x000a prepare return
    f96e: 81 4e 0c 00   mov r14,  12(r1)  ; 0x000c prepare return
    f972: 81 4f 00 00   mov r15,  0(r1) ;          prepare return
    f976: 84 12         call  r4    ;              call TimerLap()

0000f846 <mymul_hw>:
    f846: 82 4c 40 01   mov r12,  &0x0140 ;         write into memmap reg
    f84a: 82 4d 42 01   mov r13,  &0x0142 ;         write into memmap reg
    f84e: 82 4e 44 01   mov r14,  &0x0144 ;         write into memmap reg
    f852: 82 4f 46 01   mov r15,  &0x0146 ;         write into memmap reg
    f856: 3c 40 50 01   mov #336, r12 ;#0x0150
    f85a: 9c 43 00 00   mov #1, 0(r12)  ;r3 As==01  write into memmap reg
    f85e: 8c 43 00 00   mov #0, 0(r12)  ;r3 As==00  write into memmap reg
    f862: 1c 42 48 01   mov &0x0148,r12 ;0x0148     read from memmap reg
    f866: 1d 42 4a 01   mov &0x014a,r13 ;0x014a     read from memmap reg
    f86a: 1e 42 4c 01   mov &0x014c,r14 ;0x014c     read from memmap reg
    f86e: 1f 42 4e 01   mov &0x014e,r15 ;0x014e     read from memmap reg
    f872: 30 41         ret
```